An Efficient Causal Ordering Algorithm for Mobile Computing Environments

Ravi Prakash, Michel Raynal, Mukesh Singhal

▶ **To cite this version:**

Ravi Prakash, Michel Raynal, Mukesh Singhal. An Efficient Causal Ordering Algorithm for Mobile Computing Environments. [Research Report] RR-2680, INRIA. 1995. inria-00074011

HAL Id: inria-00074011
https://inria.hal.science/inria-00074011

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Efficient Causal Ordering Algorithm for Mobile Computing Environments

Ravi Prakash , Michel Raynal , Mukesh Singhal

N° 2680

Octobre 1995

PROGRAMME 1

ISSN 0249-6399

An Efficient Causal Ordering Algorithm for Mobile Computing Environments

Ravi Prakash* , Michel Raynal** , Mukesh Singhal***

Programme 1 — Architectures parallèles, bases de données, réseaux
et systèmes distribués
Projet Adp

Rapport de recherche n° 2680 — Octobre 1995 — 30 pages

Abstract: Causal message ordering is required for several distributed applications. In order to preserve causal ordering, only direct dependency information between messages, with respect to the destination process(es), should be sent with each message. By eliminating other kinds of control information from the messages, the communication overheads can be significantly reduced. In this paper we present an algorithm that uses this knowledge to efficiently enforce causal ordering of messages. The proposed algorithm does not require any prior knowledge of the network or communication topology. As computation proceeds, it acquires knowledge of the logical communication topology and is capable of handling dynamically changing multicast communication groups. With regard to communication overheads, the algorithm is optimal for the broadcast communication case. Its energy efficiency and low bandwidth requirement make it suitable for mobile computing systems. We present a strategy that employs the algorithm for causally ordered multicasting of messages in mobile computing environments.

Key-words: Causal message ordering, direct dependency, mobile computing.

(Résumé : tsvp)

*Dept. of Computer and Info. Science, The Ohio State University, Columbus, OH 43210, USA. prakash@cis.ohio-state.edu
**IRISA, Campus de Beaulieu, Rennes Cedex, France. michel.raynal@irisa.fr
***Dept. of Computer and Info. Science, The Ohio State University, Columbus, OH 43210, USA. singhal@cis.ohio-state.edu

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Un protocole d'ordonnancement causal pour les systèmes à agents mobiles

Résumé : L'ordonnancement causal a pour but de réduire l'asynchronisme des exécutions réparties en répercutant sur les livraisons de messages l'ordre de causalité qui préside à leurs émissions.

Cet article examine un protocole qui assure un tel ordonnancement causal. Ce protocole, fondé sur la seule utilisation de la dépendance directe entre émissions de messages, est particulièrement efficace. Motivé par les applications composées d'agents mobiles, il est peu couteux en information de contrôle.

Mots-clé : agents mobiles, dépendance causale directe, ordre causal.

1 Introduction

A distributed system is composed of a set of processors connected to each other by a communication network. The processors do not have access to a global clock and they do not have any shared memory that can be used to exchange information. Miniaturization of computers and advancements in communication technology are being reflected in the form of an increasing number of mobile processing units in a distributed system.

A mobile computing system is a distributed system consisting of a number of mobile and fixed processing units. The fixed units, henceforth referred to as Mobile Support Stations ($MSSs$), can communicate with each other through a fixed wireline network [10]. The geographical area of the mobile computing system is divided into regions called *cells* with an MSS in each cell. The mobile units (also referred to as Mobile Hosts - MHs) in a cell can open a wireless communication channel with the MSS in the cell and communicate with all the other processors in the system through this MSS. Alternatively, an MH can be connected through an access point of the fixed wireline network (referred to as a *telepoint* [7]) for communication purposes. An MH can move out of one cell and into another cell. In such a case the MSS of the old cell has to hand over the responsibilities for the MH's communication to the MSS of the new cell. This process is referred to as *hand-off*.

The wireless communication channels used by the MHs have a significantly lower bandwidth than the wireline communication links between the $MSSs$. Moreover, different telepoints in the network may provide links of different bandwidths to the MHs. Usually, the bandwidth of such links is less than the communication bandwidth available between the $MSSs$. Hence, control information sent with the messages should be kept as small as possible. This will help in minimizing the communication delays over the low bandwidth channels. Yet another motivation for efficient communication protocols is the limited energy source of MHs. Wireless communication, especially message send, drains their batteries. The longer the message, the greater the energy consumption. Moreover, the limited memory available at the MHs requires that the data structures associated with the communication protocols be as concise as possible.

A distributed application executing in a mobile computing environment consists of a collection of processes such that one or more processes may be running on each processing unit. The processes communicate with each other through asynchronous message passing (message propagation time is finite but arbitrary). The execution of a process consists of three types of events: *message send*, *message delivery*, and *internal events*. Internal events represent local computations at the processes. In the

absence of a global clock, determining the relative order of occurrence of events on different processes is non-trivial.

However, a *cause and effect* relationship, also referred to as *causal dependency* can be established between some events. An event occurring at a process is causally dependent on every preceding event that has occurred at that process. Causal dependencies between events on different processes are established by message communication. Such dependencies can be expressed using Lamport's *happened before* relation [11]. Two events are said to be mutually concurrent if there is no causal dependency between them. Thus, the *happened before* relation induces a partial ordering on events based on their causal dependencies.

Controlling the execution of a distributed application such that all the events are totally ordered is expensive and leads to a loss in concurrency [18]. A less severe form of ordering of message transmission and reception, called *causal ordering*, is required for a variety of applications like management of replicated data, observation of a distributed system, resource allocation, multimedia systems, and teleconferencing [2, 4, 15].

Protocols to implement causal ordering of messages have been presented in [5, 6, 14, 15, 17]. These protocols have high communication overheads. For each of these protocols (except [5] which is based on message duplication — a high communication overhead approach) the message overhead is at least $\Theta(N^2)$ integers, where N is the number of processes in the system. Hence, the protocols are not scalable. The scalability problem is especially important in a mobile computing environment due to limitations of available bandwidth, memory and energy supply, as described earlier.

Hence, there is a need for an implementation of causal ordering of messages that has low communication, computation and memory overheads. Such a requirement raises some pertinent questions. What is the minimum amount of information that each site needs to maintain, and each computation message needs to carry in order to enforce causal ordering? What is the extra information maintained by the existing algorithms? How can this extra overhead be eliminated without compromising the correctness of the algorithm? This paper addresses these issues and presents a low overhead algorithm for causal ordering of messages.

The rest of the paper is organized as follows: Section 2 contains a description of the system model and a formal definition of causal ordering. Section 3 briefly describes previous algorithms for causal ordering. Section 4 presents the motivation for the proposed algorithm and the basic idea behind it. The algorithm is presented in Section 5 and its correctness is proved in Section 6. Section 7 compares the performance of the proposed algorithm with existing algorithms and discusses

its performance for the broadcasting case and its suitability for mobile computing environments. In Section 8, an algorithm for causal multicasting of messages in a mobile computing environment is presented. Finally, the conclusions are presented in Section 9.

2 System Model

The application under consideration is composed of N processes. These processes collectively execute a distributed computation. There exists a logical communication channel between each pair of processes. A process can send a message to either one process or a group of processes (multicast communication can be implemented either by broadcasting the message to all the processes or through multiple point-to-point communications). The group of processes to which a process sends multicast messages is not fixed, i.e., a process P_i may send one multicast message to a group of processes G_1 and later another multicast message to a different group of processes G_2. Dynamic multicast groups are allowed because a process can do a multicast to any group of processes without having to form groups *a priori*.

The system does not have a global physical clock and the local clocks of the constituent processes are not perfectly synchronized. Hence, the order in which two events occur at two different processes cannot be determined solely on the basis of the local time of occurrence. However, information about the order of occurrence of events can be gathered based on the causal dependencies between them. Such dependencies can be expressed using the *happened before* relation ($\rightarrow$) between events. The *happened before* relation between events has been defined in [11] as:

- $a \rightarrow b$, if a and b are events in the same process and a occurred before b.
- $a \rightarrow b$, if a is the event of sending a message M in a process and b is the event of delivery of the same message to another process.
- If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$ (i.e., "$\rightarrow$" relation is transitive).

If $a \not\rightarrow b$ and $b \not\rightarrow a$, then a and b are said to be concurrent and represented as $a \parallel b$.

Events a, b, and c mentioned above can be either message $SEND$, message $DELIVER$ or internal events of the processes. Causal ordering of messages specifies the relative order in which two messages can be delivered to the application process.

Definition 1 *(Causal Ordering) If two messages M_1 and M_2 have the same destination and $SEND(M_1) \rightarrow SEND(M_2)$, then $DELIVER(M_1) \rightarrow DELIVER(M_2)$.*